**Timing Report**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mutex** | **Synchronized** | **Atomic Integer** |
| **1** | 2506 | 2529 | 2516 |
| **2** | 2505 | 2515 | 2509 |
| **3** | 2505 | 2525 | 2513 |
| **4** | 2504 | 2511 | 2516 |
| **5** | 2509 | 2505 | 2512 |
| **Average** | 2505.8 | 2517 | 2513.2 |
| **Slowest** | 2509 | 2529 | 2516 |
| **Fastest** | 2504 | 2505 | 2509 |

Measured in milliseconds

From my tests ran it appears that the mutex lock is the fastest synchronization method, coming in on average 8ms faster than the Atomic Integer method with the Synchronization a further 4ms slower again. As atomic integers do not use locks and do not require context switching between threads as with mutex locks so I would have assumed that they would have run quicker. Here that does not seem to be the case so there may be an issue with the code itself. The synchronized method acquires a lock for the entire methos it has been declared and releases it once the method has executed similar to how a mutex operates.